REMARKS

This amendment is responsive to the Office Action dated May 26, 2005. Applicants have canceled claims 10, 20 and 21 and amended claims 3, 5, 8, 11, 19, 22, 31, and 35. Claims 1-9, 11-19, and 22-35 are pending upon entry of this amendment.

Abstract

The Examiner objected to the abstract of the disclosure as having a typographical error. Applicants have amended the abstract of the disclosure to correct the error.

Claim Objections

The Examiner objected to claim 35 as having a typographical error. Applicants have amended claim 35 to correct the error.

Claim Rejections Under 35 U.S.C. § 102

Claims 1-4, 15-18, 26, 28, 29, and 33-35

In the Office Action, the Examiner rejected claims 1-4, 15-18, 26, 28, 29, and 33-35 under 35 U.S.C. 102(e) as being anticipated by Joyce (USPN 6,519,703). Applicants respectfully traverse the rejection to the extent that it remains applicable to the amended claims. Joyce fails to disclose each and every feature of the claimed invention, as required by 35 U.S.C. 102(e), and provides no teaching that would have suggested the desirability of modification to include such features.

For example, with respect to claims 1 and 26, Joyce fails to teach or suggest selectively processing a packet using a *software process or an interrupt-driven service routine*. Claims 1 and 26 specifically require "selectively processing a packet," i.e., selecting between a software process and an interrupt-driven service routine to process an inbound packet. Joyce fails to teach or suggest such a selection.

Similarly, Joyce fails to teach or suggest selecting between a first mode of processing inbound packets using interrupt service routines and a second mode of processing the inbound packets using a software process, as recited by Applicants' claim 15.

With respect to these elements, the Examiner impermissibly substituted "packets rated as 'high confidence' [that] are released into the traditional firewall rule base for further processing" and "mode," in place of the claim language, "interrupt-driven service routine." This substitution is impermissible, and the Examiner's substitute language is different from the plain language of the claims.

For these reasons, it appears that a review of the present application is necessary to assist the Examiner's understanding of the claimed invention. In general, Applicants point out that the claimed invention relates to controlling the processing of packets <u>after</u> an event is detected, such a as detection of a network attack or a determination of a current level of traffic. The processing of packets is selectively controlled, for example, to reduce the effects of the event. In contrast, the art cited by the Examiner generally relates to techniques for detecting a network attack.

Further, it is important for the Examiner to understand the difference between a "software process" and an "interrupt-driven service routine," as used within the present application and well known by a person having ordinary skill in the art.

In general, a *software process* includes one or more threads of execution including variables and other states associated with the threads. A multitasking operating system, such as Unix, switches between software processes to give the appearance of simultaneous execution, though only one process can be executing at once on a CPU core.[1] The operating system divides processor time among running software processes. This is referred to as a context switch. At each context switch, the processor gracefully saves the state of the currently executing proces and switches to another process. Execution is deterministic in that a context switch occurs at a well defined, controlled rate. For example, an operating system may switch between processes at a 10 ms interval. As evidence of this point, Tanenbaum explains the fundamental notion of software process in <u>Operating Systems: Design and Implementation</u> as follows:

> In [the process] model, all the runnable software on the computer, often including the operating system, is organized into a number of sequential processes, or just processes for short. A process is just an executing program, including the current values of the program counter, registers, and variables. Conceptually, each process has its own virtual CPU. In reality, of course, the real CPU switches back and forth from process to process, but to understand

[1] http://en.wikipedia.org/wiki/Process

> the system, it is much easier to think about a collection of processes running in (pseudo) parallel, than to try to keep track of how the CPU switches from program to program. . . .
>
> The key idea here is that a process is an activity of some kind. It has a program, input, output, and a state. A single processor may be shared among several processes, with some scheduling algorithm being used to determine when to stop work on one process and service a different one.[2]

In contrast, an interrupt-driven service routine allows a device to stop the CPU's synchronous execution of software processes in order to handle an asynchronous event. When a processor receives an interrupt, the normal flow of all software processes stops and control jumps to a concise software module called an *interrupt service routine*. The interrupt service routine is a specially written piece of software that handles the interrupt.[3] Once the asynchronous event is handled, execution returns to the previously executing software process. As one example, network packets may be asynchronously processed by an interrupt service routine after a network controller sends an interrupt to the CPU. Silberschatz et al. describe the basic notion of interrupt handling in Operating System Concepts as follows:

> When the CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location. The fixed location usually contains the starting address where the service routine for the interrupt is located. The interrupt service routine transfers data from the local buffer of the device controller to main memory. Once this transfer is accomplished, the CPU can then resume the interrupted computation. In this way, I/O devices and the CPU can be operated concurrently. . . .
>
> Interrupts are an important part of a computer architecture. Each computer design has its own interrupt mechanism, but several functions are common. The interrupt must transfer control to the interrupt service routine. Generally, this is done through reservation of a set of locations in low memory (the first 100 or so locations) to hold the addresses of the interrupt service routines for the various devices. This array, or *vector* of addresses is then indexed by a unique device number given with the interrupt to provide the address of the interrupt service routine for the interrupting device.[4]

---

[2] Tanenbaum, Andrew S. Operating Systems: Design and Implementation. Prentice-Hall, Inc., 1987, pp. 46-7.
[3] http://en.wikipedia.org/wiki/Interrupt
[4] Silberschatz, A. et al. Operating System Concepts. 3rd Ed., Addison-Wesley Publishing Company, 1991, pp. 30-1.

The information that contains the destination to which the CPU jumps for a given interrupt is termed the interrupt vector. Some computer system designs incorporate a list of such vectors in memory; this is termed the interrupt vector table or *table of pointers*.[5]

With regard to the elements of claims 1, 15, and 26, the Examiner stated that Joyce discloses "a routing engine to selectively process the packet using a heuristic stage (software process/mode) or an interrupt driven service routine (mode) based on the detection of a network attack (col. 2, lines 30-57)." However, this passage of Joyce describes a firewall that processes data packets using different firewall <u>rule sets</u> depending on a degree of confidence in the security of the data packets. Contrary to the Examiner's assertion, Joyce makes no mention of selecting between an interrupt-driven service routine and a software process, as recited in Applicants' claims 1, 15, and 26. The Examiner's substitution of "mode" for the claim elements "interrupt driven service routine" is erroneous and contrary to usage of the terms by one of ordinary skill, as explained above.

Joyce also fails to teach or suggest selectively processing a packet *based on detection of an event* or *based on detection of a network attack*, as recited by claims 1, 2, 16, and 26. The Examiner states that the event being detected by the method and apparatus disclosed in Joyce is a network attack. However, Joyce teaches that a packet should be selectively processed using different rules in order to identify a network attack. Thus, to the extent this is a form of selection, the selection occurs <u>before</u> detection of a network attack for the purpose of detecting an attack. Joyce teaches a method and apparatus for detecting a network attack, while Applicants' claims are directed to dynamically controlling packet processing <u>in response to</u> an event, such as the previous detection of a network attack.

With regard to the elements of claim 3, Joyce fails to teach or suggest invoking a service routine using a software interrupt when an event is not detected and invoking a software process using a wakeup signal upon detecting the event.

With regard to the elements of claim 28, Joyce fails to teach or suggest a routing device having a detection module comprising a network service routine invoked in response to a hardware interrupt from the network interface.

---

[5] http://en.wikipedia.org/wiki/Interrupt_vector

With regard to the elements of claim 29, Joyce fails to teach or suggest a routing device comprising a set of packet service routines to service inbound packets in accordance with a plurality of network protocols and an operating system to invoke one of the packet service routines to invoke the software process in response to a software interrupts when the network attack is not detected.

With regard to the elements of claim 35 as amended, Joyce fails to teach a routing device comprising a table of pointers to packet service routines. Moreover, the Examiner impermissibly substituted his own language, "determines the mode of operation based on the severity of the packet rating," in place of the language of claim 35, "table of pointers." In fact, the table of pointers contains the memory addresses of the interrupt service routines that process data packets.

Claims 5-14 and 19-25

In the Office Action, the Examiner rejected claims 5-14 and 19-25 under 35 U.S.C. 102(e) as being anticipated by Gleichauf et al. (Gleichauf) (USPN 6,301,668). Applicants respectfully traverse the rejection to the extent that it remains applicable to the amended claims. Gleichauf fails to disclose each and every feature of the claimed invention, as required by 35 U.S.C. 102(e), and provides no teaching that would have suggested the desirability of modification to include such features.

For example, with respect to claims 5 and 19 as amended, Gleichauf fails to teach or suggest setting a rate limiting operating mode based on a traffic level of inbound packets and selectively invoking a packet service routine from a software process. Moreover, with respect to claims 5 and 19, Gleichauf fails to teach or suggest selectively invoking a packet service routine based on the rate-limiting operating mode by calling the packet service routine from the software process without issuing an interrupt when the traffic level of the inbound packets exceeds a threshold, and issuing a software interrupt to invoke the packet service routine as an interrupt-driven service routine when the traffic level of the inbound packets does not exceed the threshold.

Gleichauf is silent with respect to these elements. In general, Gleichauf describes an intrusion detection system, i.e., a system for detecting network attacks. The portion of Gleichauf

cited by the Examiner merely describes using different attack signatures to detect different types of attacks. Gleichauf describes prioritizing the execution time allocated to the attack signatures, but this only means that the different signatures receive more or less of the processor execution cycles. This does not teach or suggest rate-limiting the total number of packets processed by a device. Moreover, this fails to disclose selection between using *a software process without issuing an interrupt* when traffic levels exceed a threshold and using an *interrupt driven software routine* when levels are below a threshold. Gleichauf makes no such distinction.

With regard to the elements of claim 6, Gleichauf fails to teach or suggest determining an execution period that the software process has executed without a context switch. The Examiner agrees that "a context switch is not used within the teachings of Gleachauf et al since it is not disclosed." Given this acknowledgement, Applicants are confused why the Examiner believed that claim 6 was anticipated by Gleichauf.

With regard to the elements of claim 7, Gleichauf fails to teach or suggest pausing execution of a software process for a sleep period when the execution period exceeds a threshold. The Examiner cited a teaching of Gleichauf to disable certain functions performed by a software process, but Gleichauf does not disclose pausing execution of a software process for a sleep period.

With regard to the elements of claim 8, Gleichauf fails to teach or suggest dynamically adjusting a sleep period during a network attack. The Examiner cited a teaching of Gleichauf to disable and enable tasks and system services according to processor utilization, but Gleichauf does not disclose adjusting a sleep period.

With regard to the elements of claims 9 and 20, Gleichauf fails to teach or suggest invoking a *packet service routine* from a software process. In fact, Gleichauf does not even discuss a packet service routine.

With regard to the elements of claims 11 and 22 as amended, Gleichauf fails to teach or suggest selecting a pointer *to invoke packet service routines* from a table of pointers *in response to an interrupt.* The mode of operation of Gleichauf is unrelated to invoking interrupt service routines.

In order to support an anticipation rejection under 35 U.S.C. 102(e), it is well established that a prior art reference must disclose each and every element of a claim. This well known rule

of law is commonly referred to as the "all-elements rule."[6] If a prior art reference fails to disclose any element of a claim, then rejection under 35 U.S.C. 102(e) is improper.[7]

Joyce fails to disclose each and every limitation set forth in claims 1-4, 15-18, 26, 28, 29, and 33-35. Gleichauf fails to disclose each and every limitation set forth in claims 5-14 and 19-25. For at least these reasons, the Examiner has failed to establish a prima facie case for anticipation of Applicants' claims 1-26, 28, 29, and 33-35 under 35 U.S.C. 102(e). Withdrawal of these rejections is requested.

Claim Rejection Under 35 U.S.C. § 103

In the Office Action, the Examiner rejected claims 27 and 30-32 under 35 U.S.C. 103(a) as being unpatentable over Joyce (USPN 6,519,703) in view of Gleichauf et al. (USPN 6,301,668). Applicants respectfully traverse the rejection. The applied references fail to disclose or suggest the inventions defined by Applicants' claims, and provide no teaching that would have suggested the desirability of modification to arrive at the claimed invention.

With reference to independent claim 27, for example, the applied references lack any teaching that would have suggested using a counter to indicate a number of packets processed for a network protocol. Neither Joyce nor Gleichauf et al. teaches a detection module that includes a counter that enables rate-limiting operating mode None of the references teaches or suggests a "rate-limiting operating mode" or a counter for use in enabling such a mode. The portions of Gleichauf cited by the Examiner do not refer to rate limiting, but the use of different attack detection signatures to be applied to traffic to detect different matching traffic patterns. Gleichauf describes prioritizing the execution time allocated to the attack signatures, but this only means that the signatures receive more or less of the processor execution cycles. This does not teach or suggest rate-limiting the total number of packets processed by a device.

In regard to claim 31, the Examiner proposes to modify the firewall rule set selection device of Joyce with the enabling and disabling of analysis tasks and system services taught by

---

[6] *See Hybritech Inc. v. Monoclonal Antibodies, Inc.*, 802 F.2d 1367, 231 USPQ 81 (CAFC 1986) ("it is axiomatic that for prior art to anticipate under 102 it has to meet every element of the claimed invention").

[7] *Id. See also Lewmar Marine, Inc. v. Barient, Inc.* 827 F.2d 744, 3 USPQ2d 1766 (CAFC 1987); *In re Bond*, 910 F.2d 831, 15 USPQ2d 1566 (CAFC 1990); *C.R. Bard, Inc. v. MP Systems, Inc.*, 157 F.3d 1340, 48 USPQ2d 1225 (CAFC 1998); *Oney v. Ratliff*, 182 F.3d 893, 51 USPQ2d 1697 (CAFC 1999); *Apple Computer, Inc. v. Articulate Systems, Inc.*, 234 F.3d 14, 57 USPQ2d 1057 (CAFC 2000).

Gleichauf et al. The Examiner again appears to concede that neither Joyce nor Gleichauf et al. teaches determining an execution period that a software process has executed without a context switch. For example, the Examiner clearly stated that "a context switch is not used within the teachings of Gleichauf et al since it is not disclosed." Thus, the Applicants are confused as to how the Examiner rejected claim 31 based on this admission.

In regard to claim 32, the Examiner proposed to modify the firewall rule set selection device of Joyce with the enabling and disabling of analysis tasks and system service taught by Gleichauf et al. However, as discussed above, Gleichauf et al. fails to disclose dynamically adjusting a sleep period during a network attack. Therefore, Gleichauf et al. and Joyce, taken together, do not teach each and every limitation set forth in claim 32.

For at least these reasons, the Examiner has failed to establish a prima facie case for non-patentability of Applicants' claims 27 and 30-32 under 35 U.S.C. 103(a). Withdrawal of this rejection is requested.

CONCLUSION

All claims in this application are in condition for allowance. Applicants respectfully request reconsideration and prompt allowance of all pending claims. Please charge any additional fees or credit any overpayment to deposit account number 50-1778. The Examiner is invited to telephone the below-signed attorney to discuss this application.

Date: September 26, 2005

By: Kent J. Sieffert

SHUMAKER & SIEFFERT, P.A.
8425 Seasons Parkway, Suite 105
St. Paul, Minnesota 55125
Telephone: 651.735.1100
Facsimile: 651.735.1102

Name: Kent J. Sieffert
Reg. No.: 41,312